N 11

```
VVV          VVV  MMM          MMM     SSSSSSSSSSSSS   LLL                  IIIIIIIII      BBBBBBBBBBBBB
VVV          VVV  MMM          MMM     SSSSSSSSSSSSS   LLL                  IIIIIIIII      BBBBBBBBBBBBB
VVV          VVV  MMM          MMM     SSSSSSSSSSSSS   LLL                  IIIIIIIII      BBBBBBBBBBBBB
VVV          VVV  MMMMMM    MMMMMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMMMMM    MMMMMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMMMMM    MMMMMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMM   MMM    MMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMM   MMM    MMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMM   MMM    MMM  SSS                LLL                     III         BBB          BBB
VVV          VVV  MMM          MMM     SSSSSSSSSS      LLL                     III         BBBBBBBBBBBBB
VVV          VVV  MMM          MMM     SSSSSSSSSS      LLL                     III         BBBBBBBBBBBBB
VVV          VVV  MMM          MMM     SSSSSSSSSS      LLL                     III         BBBBBBBBBBBBB
VVV          VVV  MMM          MMM               SSS   LLL                     III         BBB          BBB
VVV          VVV  MMM          MMM               SSS   LLL                     III         BBB          BBB
VVV          VVV  MMM          MMM               SSS   LLL                     III         BBB          BBB
   VVV    VVV     MMM          MMM               SSS   LLL                     III         BBB          BBB
   VVV    VVV     MMM          MMM               SSS   LLL                     III         BBB          BBB
   VVV    VVV     MMM          MMM               SSS   LLL                     III         BBB          BBB
      VVV         MMM          MMM  SSSSSSSSSSSSS      LLLLLLLLLLLLLLL      IIIIIIIII      BBBBBBBBBBBBB
      VVV         MMM          MMM  SSSSSSSSSSSSS      LLLLLLLLLLLLLLL      IIIIIIIII      BBBBBBBBBBBBB
      VVV         MMM          MMM  SSSSSSSSSSSSS      LLLLLLLLLLLLLLL      IIIIIIIII      BBBBBBBBBBBBB
```

```
**FILE**ID**EODEFM                                        D 11



          EEEEEEEEEE    000000    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  MM      MM
          EEEEEEEEEE    000000    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  MM      MM
          EE          OO      OO  DD      DD  EE          FF          MMMM  MMMM
          EE          OO      OO  DD      DD  EE          FF          MMMM  MMMM
          EE          OO      OO  DD      DD  EE          FF          MM  MM  MM
          EE          OO      OO  DD      DD  EE          FF          MM  MM  MM
          EEEEEEEE    OO      OO  DD      DD  EEEEEEEE    FFFFFFFF    MM      MM
          EEEEEEEE    OO      OO  DD      DD  EEEEEEEE    FFFFFFFF    MM      MM
          EE          OO      OO  DD      DD  EE          FF          MM      MM
          EE          OO      OO  DD      DD  EE          FF          MM      MM
          EE          OO      OO  DD      DD  EE          FF          MM      MM      ....
          EE          OO      OO  DD      DD  EE          FF          MM      MM      ....
          EEEEEEEEEE    000000    DDDDDDDD    EEEEEEEEEE  FF          MM      MM      ....
          EEEEEEEEEE    000000    DDDDDDDD    EEEEEEEEEE  FF          MM      MM      ....


          MM      MM    AAAAAA    RRRRRRRR
          MM      MM    AAAAAA    RRRRRRRR
          MMMM  MMMM  AA      AA  RR      RR
          MMMM  MMMM  AA      AA  RR      RR
          MM  MM  MM  AA      AA  RR      RR
          MM  MM  MM  AA      AA  RR      RR
          MM      MM  AA      AA  RRRRRRRR
          MM      MM  AA      AA  RRRRRRRR
          MM      MM  AAAAAAAAAA  RR  RR
          MM      MM  AAAAAAAAAA  RR  RR
          MM      MM  AA      AA  RR    RR
          MM      MM  AA      AA  RR    RR
          MM      MM  AA      AA  RR      RR
          MM      MM  AA      AA  RR      RR
```

```
        .TITLE EODEF - EDITPC Pattern Operator Macros
        .IDENT 'V04-000'
        .NLIST
;+
;****************************************************************************
;*                                                                          *
;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;*  ALL RIGHTS RESERVED.                                                    *
;*                                                                          *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;*  TRANSFERRED.                                                            *
;*                                                                          *
;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;*  CORPORATION.                                                            *
;*                                                                          *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;*                                                                          *
;*                                                                          *
;****************************************************************************
;
; FACILITY :
;       EDITPC Pattern Operator Encodings
;
; ABSTRACT :
;       The EDITPC instruction, Edit Packed to Character String, performs the
;       editing according to the pattern string which consists of one byte
;       pattern operators.  Some pattern operators take no operands.  Some
;       take a repeat count which is contained in the rightmost nibble of the
;       pattern operator itself.  The rest take a one byte operand which
;       follows the pattern operator immediately.  This operand is either an
;       unsigned integer length or byte character.  Edit patterns must end
;       with the EO$END pattern operator.
;
;       The EODEF macros permit easy construction of the edit pattern.
;
; ENVIRONMENT:
;
; AUTHOR:
;       R. P. Grosso,           Creation Date 9-Oct-1980
;
; MODIFIED BY:
;
;-
```

```
; EO$END EDIT PATTERN OPERATOR ENCODING
;       CALL EO$END
;
        .MACRO  EO$END
        .BYTE   ^X00                    ; EO$END encoding
        .ENDM   EO$END

; EO$END_FLOAT EDIT PATTERN OPERATOR ENCODING
;       CALL EO$END_FLOAT
;
        .MACRO  EO$END_FLOAT
        .BYTE   ^X01                    ; EO$END_FLOAT encoding
        .ENDM   EO$END_FLOAT

; EO$CLEAR_SIGNIF EDIT PATTERN OPERATOR ENCODING
;       CALL EO$CLEAR_SIGNIF
;
        .MACRO  EO$CLEAR_SIGNIF
        .BYTE   ^X02                    ; EO$CLEAR_SIGNIF encoding
        .ENDM   EO$CLEAR_SIGNIF

; EO$SET_SIGNIF EDIT PATTERN OPERATOR ENCODING
;       CALL EO$SET_SIGNIF
;
        .MACRO  EO$SET_SIGNIF
        .BYTE   ^X03                    ; EO$SET_SIGNIF encoding
        .ENDM   EO$SET_SIGNIF

; EO$STORE_SIGN EDIT PATTERN OPERATOR ENCODING
;       CALL EO$STORE_SIGN
;
        .MACRO  EO$STORE_SIGN
        .BYTE   ^X04                    ; EO$STORE_SIGN encoding
        .ENDM   EO$STORE_SIGN

; EO$LOAD_FILL EDIT PATTERN OPERATOR ENCODING
;       CALL EO$LOAD_FILL <CH>
;       WHERE CH IS THE FILL CHARACTER
;
        .MACRO  EO$LOAD_FILL CH
        .BYTE   ^X40                    ; EO$LOAD_FILL encoding
        .IF BLANK <CH>                  ; check to be sure CH isn't blank
        .WARN           ;EO$LOAD_FILL - CHARACTER WAS BLANK OR NOT DELIMITED
        .ENDC
        .IF IDENTICAL CH,/              ; avoid .ASCII ///       if CH is /
        .BYTE   ^X2F                    ; enter ASCII for "/"
        .IF_FALSE                       ; if CH is not "/" then
        .ASCII  /CH/                    ; fill char placed in fill register
        .ENDC
        .ENDM   EO$LOAD_FILL
```

```
; EO$LOAD_SIGN EDIT PATTERN OPERATOR ENCODING
;       CALL EO$LOAD_SIGN <CH>
;       WHERE CH IS THE SIGN CHARACTER
;
        .MACRO  EO$LOAD_SIGN CH
        .BYTE    ^X41                   ; EO$LOAD_SIGN  encoding
        .IF BLANK <CH>                  ; check to be sure CH isn't blank
        .WARN           ;EO$LOAD_SIGN - CHARACTER WAS BLANK OR NOT DELIMITED
        .ENDC
        .IF IDENTICAL CH,/              ; avoid .ASCII ///      if CH is /
        .BYTE    ^X2F                   ; enter ASCII for "/"
        .IF FALSE                       ; if CH is not "/" then
        .ASCII  /CH/                    ; sign char placed in sign register
        .ENDC
        .ENDM   EO$LOAD_SIGN

; EO$LOAD_PLUS EDIT PATTERN OPERATOR ENCODING
;       CALL EO$LOAD_PLUS <CH>
;       WHERE CH IS THE SIGN CHARACTER WHEN RESULT IS POSITIVE
;
        .MACRO  EO$LOAD_PLUS CH
        .BYTE    ^X42                   ; EO$LOAD_PLUS  encoding
        .IF BLANK <CH>                  ; check to be sure CH isn't blank
        .WARN           ;EO$LOAD_PLUS - CHARACTER WAS BLANK OR NOT DELIMITED
        .ENDC
        .IF IDENTICAL CH,/              ; avoid .ASCII ///      if CH is /
        .BYTE    ^X2F                   ; enter ASCII for "/"
        .IF FALSE                       ; if CH is not "/" then
        .ASCII  /CH/                    ; char to be placed in sign register
        .ENDC
        .ENDM   EO$LOAD_PLUS

; EO$LOAD_MINUS EDIT PATTERN OPERATOR ENCODING
;       CALL EO$LOAD_MINUS <CH>
;       WHERE CH IS THE SIGN CHARACTER WHEN RESULT IS NEGATIVE
;
        .MACRO  EO$LOAD_MINUS CH
        .BYTE    ^X43                   ; EO$LOAD_MINUS encoding
        .IF BLANK <CH>                  ; check to be sure CH isn't blank
        .WARN           ;EO$LOAD_MINUS - CHARACTER WAS BLANK OR NOT DELIMITED
        .ENDC
        .IF IDENTICAL CH,/              ; avoid .ASCII ///      if CH is /
        .BYTE    ^X2F                   ; enter ASCII for "/"
        .IF FALSE                       ; if CH is not "/" then
        .ASCII  /CH/                    ; char to be placed in sign register
        .ENDC
        .ENDM   EO$LOAD_MINUS

; EO$INSERT EDIT PATTERN OPERATOR ENCODING
;       CALL EO$INSERT <CH>
;       WHERE CH IS INSERTED
;
```

```
        .MACRO  EO$INSERT CH
        .BYTE   ^X44                    ; EO$INSERT encoding
        .IF BLANK <CH>                  ; check to be sure CH isn't blank
        .WARN           ;EO$INSERT - CHARACTER WAS BLANK OR NOT DELIMITED
        .ENDC
        .IF IDENTICAL CH,/              ; avoid .ASCII ///      if CH is /
        .BYTE   ^X2F                    ; enter ASCII for "/"
        .IF_FALSE                       ; if CH is not "/" then
        .ASCII  /CH/                    ; char to be inserted
        .ENDC
        .ENDM   EO$INSERT

; EO$BLANK_ZERO EDIT PATTERN OPERATOR ENCODING
;       CALL EO$BLANK_ZERO LEN
;       WHERE LEN IS A POSITIVE INTEGER
;
        .MACRO  EO$BLANK_ZERO LEN
        .BYTE   ^X45                    ; EO$BLANK_ZERO encoding
        .IF EQUAL LEN
        .WARN   ;EO$BLANK_ZERO - LENGTH SHOULD NOT EQUAL ZERO
        .ENDC
        .BYTE   LEN                     ; length to fill with contents of
                                        ;  fill register if value of source
                                        ;  string is zero.
        .ENDM   EO$BLANK_ZERO



; EO$REPLACE_SIGN EDIT PATTERN OPERATOR ENCODING
;       CALL EO$REPLACE_SIGN LEN
;       WHERE LEN IS A POSITIVE INTEGER
;
        .MACRO  EO$REPLACE_SIGN LEN
        .BYTE   ^X46                    ; EO$REPLACE_SIGN encoding
        .IF EQUAL LEN
        .WARN   ;EO$REPLACE_SIGN - LENGTH SHOULD NOT EQUAL ZERO
        .ENDC
        .BYTE   LEN
        .ENDM   EO$REPLACE_SIGN


; EO$ADJUST_INPUT EDIT PATTERN OPERATOR ENCODING
;       CALL EO$ADJUST_INPUT LEN
;       WHERE LEN IS A POSITIVE INTEGER
;
        .MACRO  EO$ADJUST_INPUT LEN
        .BYTE   ^X47                    ; EO$ADJUST_INPUT encoding
        .IF LESS_EQUAL LEN
        .WARN   ;EO$ADJUST_INPUT - LENGTH MUST BE GREATER THAN ZERO
        .ENDC
        .IF GREATER_EQUAL LEN - 32
        .WARN   ;EO$ADJUST_INPUT - LENGTH MUST BE LESS THAN 32
        .ENDC
```

```
JP

PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
PH

;
JP
```

```
        .BYTE   LEN                     ; length to fill with contents of
                                        ;  fill register if value of source
                                        ;  string is zero.
        .ENDM   EO$ADJUST_INPUT

; EO$FILL EDIT PATTERN OPERATOR ENCODING
;       CALL EO$FILL R
;       WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE
;
        .MACRO  EO$FILL R
        .IF LESS_EQUAL R                ; repeat cannot be zero
        .WARN   ;EO$FILL - REPEAT CANNOT BE ZERO
        .ENDC
        .IF GREATER_EQUAL R - 16        ; repeat count must be contained in
                                        ;  a nibble
        .WARN   ;EO$FILL - REPEAT IS GREATER THAN 15
        .ENDC
        .BYTE   <^X80 + R>              ; EO$FILL operator encoding
                                        ;  plus repeat count
        .ENDM   EO$FILL

; EO$MOVE EDIT PATTERN OPERATOR ENCODING
;       CALL EO$MOVE R
;       WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE
;
        .MACRO  EO$MOVE R
        .IF LESS_EQUAL R                ; repeat cannot be zero
        .WARN   ;EO$MOVE - REPEAT CANNOT BE ZERO
        .ENDC
        .IF GREATER_EQUAL R - 16        ; repeat count must be contained in
                                        ;  a nibble
        .WARN   ;EO$MOVE - REPEAT IS GREATER THAN 15
        .ENDC
        .BYTE   <^X90 + R>              ; EO$MOVE operator encoding
                                        ;  plus repeat count
        .ENDM   EO$MOVE

; EO$FLOAT EDIT PATTERN OPERATOR ENCODING
;       CALL EO$FLOAT R
;       WHERE R, THE REPEAT COUNT IS BETWEEN 1 AND 15 INCLUSIVE
;
        .MACRO  EO$FLOAT R
        .IF LESS_EQUAL R                ; repeat cannot be zero
        .WARN   ;EO$FLOAT - REPEAT CANNOT BE ZERO
        .ENDC
        .IF GREATER_EQUAL R - 16        ; repeat count must be contained in
                                        ;  a nibble
        .WARN   ;EO$FLOAT - REPEAT IS GREATER THAN 15
        .ENDC
        .BYTE   <^XA0 + R>              ; EO$FLOAT operator encoding
                                        ;  plus repeat count
        .ENDM   EO$FLOAT
```

```
;+
;       END OF EDIT PATTERN OPERATOR ENCODINGS
;
;-
```

0434 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

CLIMAC
REQ

TPAMAC
REQ

UTLDEFB
B32

STARMISC
MAR

EODEFM
MAR

TPAMAR
MAR

CALBYNAME
LIS

SCRPROLOG
REQ

STARLET
SDL

SCRTCB
REQ

OFFSET
MAR

SYITABLE
MAR

EODEFB
B32

JPITABLE
MAR

DVITABLE
MAR

UTLDEFM
MAR

B32MSG
LIS

SCRTERM
REQ

RSXLBLDF
MAR

C74MSG
LIS